Australian Government
Department of Defence
Defence Science and
Technology Organisation

DSTO

XQuery Engine Prototype

Egon Kuster and Andrew Roff

DSTO-TN-0577

Australian Government
Department of Defence
Defence Science and
Technology Organisation

XQuery Engine Prototype

Coalition Theatre Logistics (CTL)
Advanced Concept Technology Demonstrator (ACTD)

Egon Kuster[+] *and Andrew Roff*[*]

[+]**Command and Control Division**
Information Sciences Laboratory

[*]**Command and Control Division vacation student**

TECHNICAL NOTE
DSTO-TN-0577

Abstract

During the architectural design of the Coalition Theatre Logistics (CTL) Advanced Concept Technology Demonstrator (ACTD) it was identified that a data query capability that operates over XML-based web services was required. This document outlines how a new technology called XQuery could provide this XML-based query capability over web service oriented communication. Also outlined is a possible solution along with a discussion of its limitations and capabilities. A successful implementation of the XQuery engine was developed with performance metrics and architectural designs of the implemented system included within.

RELEASE LIMITATION

Approved for public release.

Published by

DSTO Information Sciences Laboratory
PO Box 1500
Edinburgh, South Australia, 5111
Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-013-071
November 2004

APPROVED FOR PUBLIC RELEASE.

XQuery Engine Prototype

Coalition Theatre Logistics (CTL)

Advanced Concept Technology Demonstrator (ACTD)

Executive Summary

Australia is involved in a joint project called CTL ACTD. The CTL ACTD project is responsible for researching the policy and technical issues with sharing logistic data between multiple nations that operate together in coalition operations. CTL ACTD has three main objectives:

1. **Coalition Movement Planning** – To provide the ability to plan and monitor movement of troops, equipment and supplies in and out of the Area of Operations (AO).
2. **Supply and Sustainment (with In-Transit Visibility (ITV))** – Allow the coalition task commander the ability to plan the supply and sustainment requirements of the coalition force. This objective is also striving to provide ITV feeds to track troops, equipment, and supplies in and out of the AO for plan execution monitoring.
3. **Infrastructure Visibility** – Allow the Coalition Task Force Commander the ability to view details about infrastructure required to support the logistics of the coalition operation. This includes information about roads, warehouses, airports, seaports, etc.

During the architectural design phase of CTL a number of capabilities were identified that require further analysis before integration. One such capability is a query and search component for data exposed by web services. In the CTL architecture all data communication is handled via the use of web services. Web services allow for applications to request data in the form of XML documents. Within the architecture there are numerous web services implemented providing different information or functionality. Some web services allow data to be received for storage in a database, while others allow for particular types of data to be requested. For this XQuery prototype we are only interested in the latter type of web service.

In the current CTL architecture when a client requests an XML document it calls a web service to return the required XML. For example if the client is interested in movement requests it would call the movement request web service to return the movement request XML documents. The problem with this method of operation is that the web service does not discriminate and returns all documents it can access. This can lead to extremely large results, even if the client only requires part of the data returned. To constrain the results the client needs to send its selection criteria in its initial request. XQuery could be used to define the criterion. The XQuery Prototype described in this document defines how the XQuery standard can be incorporated with the CTL web services.

The XQuery Prototype is implemented as a web service that sits on top of a database allowing the advanced XQuery searching capabilities to be applied to the XML data returned. Data contained in the database is stored in a relational structure while XML documents that XQuery operates on is hierarchical. To support the mapping between the relational and hierarchical structures an additional component was built so that the XQuery statements could be applied. The XQuery enhanced web services then operate by extracting the XML documents using the mapping component and applying the XQuery statement so only required data is returned to the client.

Testing performed on the prototype showed that the XQuery Engine is quite fast, and that the relational to hierarchical mapping transformation is the actual bottleneck in the system. More than half of the total time of a query is spent transforming the data and, when there are many records in the database, this transformation time can become unacceptably high. The tests also revealed that the transformation takes much longer with Oracle than it does with SQL Server databases.

Although the relational data to XML transformation took considerable time, this time lag is already present in the current CTL architecture. Therefore the addition of the XQuery capability provides extra functionality with similar performance.

The XQuery Prototype successfully demonstrated that XQuery could be used to search and constrain results from web services. With performance improvements the XQuery capability will prove to be a valuable capability for use in CTL or other web service deployments.

Contents

Abbreviations

ACTD	Advanced Concept Technology Demonstrator
AO	Area of Operations
API	Application Programming Interface
CIE	Coalition Information Environment
CJLOG	Commander Joint Logistics
CKO	Chief Knowledge Officer
CORBA	Common Object Request Broker Architecture
CTF	Coalition Task Force
CTL	Coalition Theatre Logistics
DOM	Document Object Model
HQ USCINCPAC	U.S. Commander-In-Chief Pacific
ITV	In-Transit Visibility
JDBC	Java Database Connectivity
JDOM	Java Document Object Model
MS	Microsoft
NRP	National Release Point
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCN	Troop Contributing Nation
W3C	World Wide Web Consortium
WG	Working Group
WSDL	Web Service Definition Language
XML	Extensible Mark-up Language

1. Introduction

In February 2000, Australia accepted a Headquarters, U.S. Commander-In-Chief, Pacific (HQ USCINCPAC J4) invitation to participate in a project to improve logistics interoperability and situational awareness between coalition partners. At the subsequent Staff Level Meeting, Under Secretary Materiel agreed to Australian partnership in a US project to be known as CTL-ACTD.

A CTL-ACTD Working Group (WG) was established in February 2001 to steer the project. Terms of Reference were signed on 25 May 2001.

The CTL-ACTD initiative was introduced to enhance combat service support to coalition forces by providing operational and logistic staff with access to timely, integrated and accurate logistic information. A suite of decision-making support tools will be included to complete the utility of the system provided.

In order to achieve the objective of generating timely and accurate coalition logistics information for the Coalition Task Force (CTF) and the respective Troop Contributing Nation (TCN) chain of commands, CTL ACTD demonstrates the following capabilities:

- Coordinated multinational system(s) and decision support tools for deployment planning and sustainment across the full spectrum of military operations.
- Information generation and dissemination tools that support situational awareness among coalition partners.
- Tools that provide multinational logistics analysis.

The combination of these capabilities addresses shortfalls in coalition logistics information sharing and decision-making.

A study[1] was conducted by both Australia and the U.S. on the coalition logistics requirements. From the comprehensive list, three high priority requirements were selected for CTL ACTD, these were:

CTLREQ1 - Coalition Movement Planning - To allow the CTF commander to plan and monitor movement of troops, equipment and supplies in and out of the Area of Operations (AO).

CTLREQ2 - Supply and Sustainment - To allow the CTF commander to plan the supply and sustainment requirements for the coalition force. This objective also covers the implementation of In-Transit Visibility (ITV) feeds for tracking personnel, equipment and supplies for plan execution monitoring.

CTLREQ3 - Infrastructure Visibility - To allow the CTF commander to view details about infrastructure required to support the logistics of the coalition operation. This includes information about roads, warehouses, airports, seaports, etc.

Many of the technologies and concepts in this document are described at a high level. It is assumed that the readers are familiar with Web Services, Relational Databases and XML. For more information about these topics please read the references for this document first.

1.1 CTL Architecture

The CTL architecture is comprised of three key components, the Coalition Server, National Release Points (NRP) and national release components (see Figure 1 below). These components are spread across two environments, the national domain that identifies the national classified network (for example SIPRNET - US National Domain) and the Coalition Information Environment (CIE), which is the coalition network. Each national domain contains national tools, systems and services used for day-to-day national operations, while the CIE contains the coalition services, tools and coalition releasable data provided by coalition nations. The CIE operates on the coalition network that provides the physical network connection between all coalition nations.

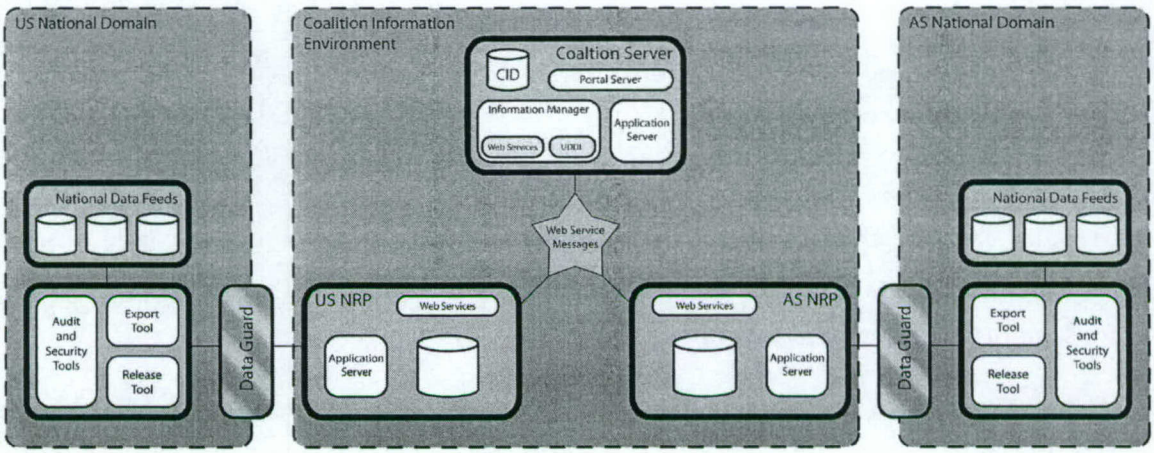


Figure 1 - CTL Architecture

The CIE contains both the coalition server and each nation's NRP. These components coupled with the web service communication layer provide the framework for sharing and distributing data and applications between nations. Each nation owns their own NRP, therefore Australia and the US would each own an NRP. Each nation is responsible for maintaining their NRP. These NRPs contain web services, application servers and databases to store coalition releasable data and applications for sharing with other coalition users. Web services are used to expose data contained in the NRP database for use by coalition applications or foreign NRPs.

The key to the data communication in the CIE is the use of standardised web services and XML documents. This allows for a common data representation to be used for all data contained or transmitted in the CIE. The web service standards define the mothod of accessing the data while the data standards define what format and type of information is delivered by the web services. The web service infrastructure exploits

the use of a component called the Information Manager contained in the Coalition Server. The information manager can broker web services transactions by collating coalition data within the CIE. As data can be stored in multiple places applications and services are required to extract the data from all these locations to get a complete data set. The Information Manager helps by providing services to execute the distributed fetch and collating of data.

The Information Manager is contained in the Coalition Server component. The Coalition Server also contains other coalition services including an application server, the Coalition Information Database (CID), File Server, Notification Server, Portal Server and Security Services. These components help tie the whole CTL CIE infrastructure together and provide the required services to view and run coalition applications. For a more detailed description of these components please refer to the CTL architecture paper[2].

For the purpose of the XQuery Engine the important components are the web services and databases. In the current version of the CTL architecture data is extracted from the databases by calling the local web service. For each database there can be many different web services, each providing different data from the database. For instance one web service may expose all the movement requests while another may expose all the equipment from the database. A shortfall of the current web services is that they do not provide any querying capability. The XQuery prototype has been developed to overcome this shortfall. The prototype explores the viability of using the XQuery language in conjunction with web services to provide query capabilities on XML documents.

1.2 CTL History

Joint Warrior Interoperability Demonstration (JWID) 2002 included the first demonstration of CTL's capabilities. This demonstration proved the concept although there were some fundamental problems with the underlying architecture. These were addressed through a rework of the CTL architecture, which was agreed at the June 2002 Technical Working Group held in Canberra, Australia.

The second half of 2002 was spent fleshing out the details of the CTL architecture and identifying shortfalls in the design. One such shortfall was the ability to query the web services to customise the eXtensible Mark-up Language (XML) data being sent. This query capability led to the development of this XQuery Engine Prototype project.

For three months, starting in December 2002, Andrew Roff worked with Egon Kuster in the development of the XQuery Engine Prototype. This document outlines the architecture, technology and results of this project. This report will also be used to better inform how the query capability can be incorporated into the overall CTL architecture.

2. XQuery Engine Structure

The XQuery Engine uses a variety of technologies to supply the querying capability on the CTL web services. This section first describes the technologies used and then how the prototype was implemented. Reasons for developing and using the selected technologies are included at the end of this section. Detailed descriptions of the technologies used is beyond this document's scope, for more details please visit the resources provided in the references section of this document.

2.1 Technologies Used

There are three relatively new technologies employed in the XQuery prototype: SOAP, JDOM and XQuery. Each is briefly described below.

2.1.1 SOAP and WSDL

SOAP (Simple Object Access Protocol) defines the message structure and client/server components for accessing and sending information between services as shown in Figure 2. SOAP is an XML based messaging protocol and is a useful tool for the exchange of information between remote, decentralised locations. SOAP messages can be simple, but because a SOAP message has an XML-like structure, it is also capable of communicating complex multi-part messages. In the XQuery prototype SOAP is used to send data requests containing the XQuery statement and respond with an XML document. For more information about the SOAP standard see reference [4].

WSDL (Web Services Description Language) provides a language for defining a web service. A WSDL document contains a description of a web services method including the input it expects and the output it will provide. The XQuery prototype's WSDL is included in appendix B of this document. For more information about WSDL see reference [5].

2.1.2 JDOM

Java Document Object Model (JDOM) is an Application Programming Interface (API) for building a Java representation of XML documents. JDOM allows the application programmer to create or manipulate XML documents using familiar java object constructs rather than the more difficult World Wide Web Consortium (W3C) Document Object Model (DOM). In the XQuery Prototype, JDOM is used to create XML documents from the relational data contained in either an Oracle of SQL Server database. For more information about JDOM see reference [6].

2.1.3 XQuery

XQuery is a query language created by the W3C as a way of searching through XML documents to find relevant information. It is an extension of XPath[7] version 2.0, with powerful searching capabilities. XQuery[8] uses XPath to select specific elements in

XML documents to either select or search for. This allows XQuery to be used to search through XML documents for specific elements or data and then customise what is returned. To learn more about the syntax of the XQuery language, refer to Appendix D. Although a relatively new standard, there are already many prototypes and implementations of XQuery available.

The XQuery Implementation used in the prototype is called XQEngine, designed by Howard Katz. This engine uses the power of XQuery to search through an XML document and output the results of the query. More information on XQEngine is available at http://www.fatdog.com/. For more detailed information about XQuery see reference [10].

2.2 XQuery Implementation

The XQuery prototype has been implemented to use both the Oracle and SQL Server databases so that speed differences between implementations could be measured. Figure 2 below outlines the implementation structure of the XQuery prototype. There are four key components, the client, the front servlet, oracle servlet and Microsoft (MS) SQL Server servlet. Each is described in more detail later.

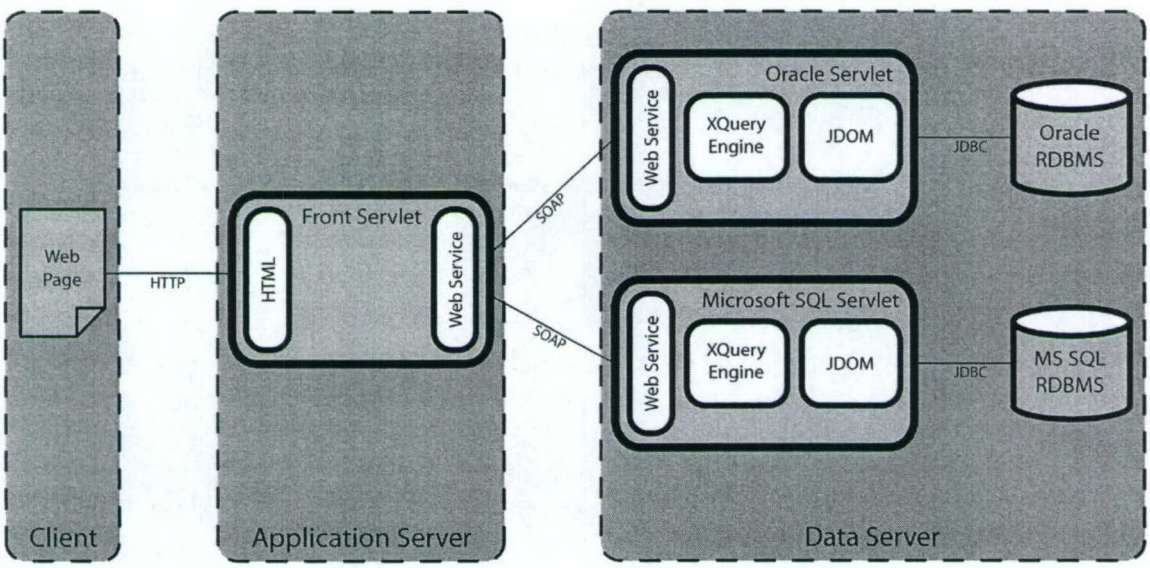


Figure 2 - Overview XQuery Engine Architecture

2.2.1 Process Flow

There are five steps involved in processing an XQuery statement on data stored in the relational databases to return the required XML results:

1. The user selects the database to be searched and inputs his or her query. This is done via the client web page.
2. The query is inserted into a SOAP message and sent to the appropriate data servlet.

3. An XML document is created from the data in the database using JDOM.
4. The data servlet extracts the query from the SOAP message and applies it to query the newly created XML document.
5. The resulting XML document is sent back to the Front servlet as a SOAP message and returned to the client.

2.2.2 Client

The client consists of a web page that allows the user to enter a query and specify a database to search. This information is sent to the application server (see Figure 2) for inclusion in a SOAP message to be sent to the Data Server. When the results of the query are sent back to the application server, it is passed back to the client and displayed in the client web browser as an XML document.

2.2.3 Front Servlet

The front servlet receives information from the client about the query and which database to search. Based on this information the servlet then sends a SOAP message containing the query to a servlet deployed on the data server (see Figure 2). The choice of servlet depends on which database the user selected on the client web page.

When the data servlet sends back the results of the query as another SOAP message, the front servlet extracts the XML results and sends it back to the client for display in the client's web browser.

The front servlet essentially acts as a broker between the client web page and the data server to convert between HTTP requests and SOAP messages.

2.2.4 Oracle/MS SQL Servlet

Whether the user selects an Oracle or SQL Server database, the execution of the query is virtually identical. The servlet receives SOAP messages from the application server and extracts the queries. Using a series of SQL queries and JDOM, the servlet generates an XML document containing all of the movement requests stored in the database. The XQuery prototype uses the CTL movement request data for sample data, a sample of this data is contained in appendix C. The resulting XML document and extracted XQuery statement is then fed into the XQEngine application, which searches the XML document using the XQuery, outputting the XML results. A SOAP message is then created to send the resulting XML file back to the application servlet.

The conversion from relational database to XML involves the creation of a series of XML elements using JDOM. The structure of these elements is dictated in the Movement Request List schema. To create an element in JDOM a series of Structured Query Language (SQL) queries are executed, the results yield the data to be stored in the element. When the construction of the JDOM XML document is completed, the result is converted into an XML string for use in the XQuery Engine.

The implementation of XQuery used in the prototype is known as XQEngine. The engine uses an XQuery query string and XML string as input and executes the query on the XML string. The result from this process is another XML document. These results are returned to the front servlet wrapped up in a SOAP message.

2.3 Reasons for Technology Choices

In the development of the XQuery prototype there were a number of technology choices. Many of the solutions selected were based on the requirement to closely align the XQuery prototype with the environment used in the CTL project. This section describes the technologies chosen and why they were selected.

2.3.1 Database

Oracle and SQL server are the two primary large Relational Database Management Systems (RDBMS) in use by both the US and Australian Departments of Defence and therefore were the natural choices for creating test databases to use with the prototype. However, with minimal adjustment, the prototype could be adapted to make use of any relational database with a JDBC (Java Database Connectivity)[11] driver. JDBC is used within the prototype to allow the data servlets to communicate with the database to run SQL queries and return the results. Most modern relational databases support the JDBC standard.

While some native XML database options were explored, it became apparent that this solution would not be suitable for all of the services within CTL that would be referencing the database. The advantage of a traditional relational database is that when data is altered, the change is reflected in all applications referencing the data. This is not necessarily possible with a native XML solution as there is no inter-dependency between data elements. Because of the relational nature of the data and that the XML documents map to overlapping tables if data changes in one XML document then this will also change data in another XML document. This is because the XML schema to relational database mapping is not mutually exclusive. This is visually represented in Figure 3 below. As shown schema 1 maps onto tables 1, 2, 3, 4 and 6 while schema 2 maps onto tables 4, 5, 6 and 7, which means that the two schemas share the use of tables 4 and 6. This is only an example and therefore does not represent the real database structure used but it does display the overlapping nature of the database to XML schema mapping used in this prototype and in the CTL system. The real database schema used in this example can be seen in Appendix A.

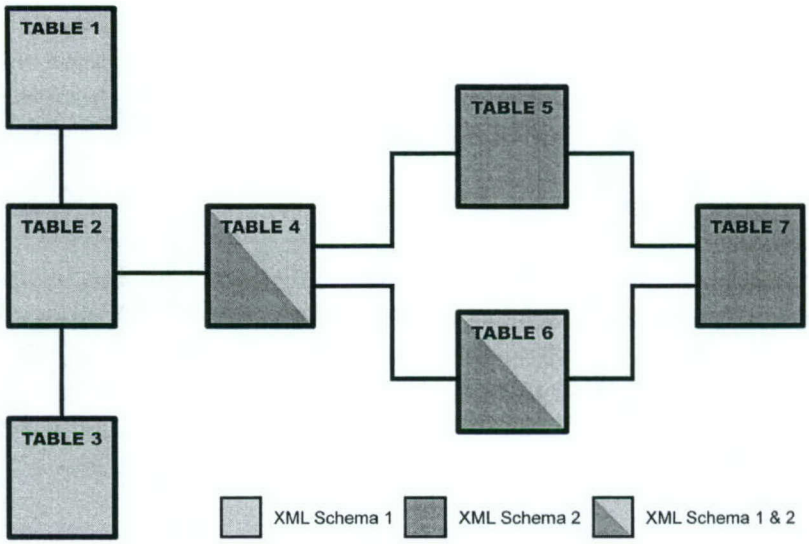


Figure 3 - XML Schema to Database Map (example only)

The overlapping nature of the data contained in the database does not allow any relational to XML mapping technologies currently available to be used, this is explored further in Section 2.3.3.

2.3.2 Messaging

SOAP is fast becoming a widely used commercial standard for transferring messages across networks and the Internet. Because SOAP messages are XML encoded, there is scope for extending the format and sending more complex and detailed messages if required in the future. SOAP is also being used for all data communication in CTL, so it will be easier to integrate code from the prototype into already existing applications. Finally, there is a large amount of support for SOAP available, with a large number of toolkits that make developing SOAP applications easier. SOAP is platform and implementation independent, which means that it can be used on any type of platform (eg. Intel, Unix, MacOS, etc.) and developed using any programming language (eg. C++, Java, C#, VB, etc.). This flexibility allows for the integration and communication between many different applications and services. Other communication methods were considered, including CORBA (Common Object Request Broker Architecture) and RMI (Remote Method Invocation) although none matched the portability and flexibility allowed with the use of SOAP.

2.3.3 Database to XML Conversion

In exploring the transformation from relational data to XML, a number of options were investigated. Both Oracle and SQL Server have their own tools for use with their databases. Without employing its object-relational capabilities, Oracle's XML SQL Utility (XSU) was only capable of producing flat (2 elements deep) XML documents. SQL Server's built in XML transformation tool is sufficiently powerful but because of the method in which the process is controlled, the code for any transformation would be incredibly complex and lengthy. This in turn would make it exceedingly difficult to modify if the structure of the database or XML schema changed. The SQL Server

approach would also have taken considerable implementation time which was unsatisfactory for this short prototype project.

The solution settled on was a combination of JDOM and many small SQL queries over JDBC connections to the database. The transformation code is relatively simple and concise, so that if there are any changes made to the database the code can be easily updated. Also, by changing the connection details, the same program can be used to query any relational database with an identical table structure. This allowed for multiple database to be used in the prototype without having to rewrite the data extraction code for each database.

2.3.4 XQuery Engine

A search of implemented XQuery engines uncovered the XQEngine software developed by a Howard Katz. The XQEngine has a number of advantages over other XQuery implementations. It is built to handle large XML documents (up to 32,000 elements, although in the next version the maximum limit is expected to be abolished) and a large number of documents (up to 32,000 documents can be searched in one query). Secondly, XQEngine is *fast*. Timing tests of the prototype have shown that the bottleneck of the process was the transformation to XML not in processing the query, this is discussed further in section 3 on the systems performance. Therefore the XQEngine provided the speed and features that were required in the XQuery prototype.

3. Performance

Performance tests were run on the prototype under a number of different conditions. Speed differences with data stored in Oracle and SQL server databases were compared as well as differing connection speeds. The computer for the databases and data servlets was a Pentium III 900 Mhz with 1024 MB RAM.

For each database, tests were run with 10, 50 and 100 records in the database. For 10 records the size of the transformed XML document was 82 KB, or 3744 lines of text. For 100 records, the transformation produced a 967 KB document, with 19,634 lines.

To measure the time the prototype took to run the system time was recorded at five stages during execution:

- When the front servlet received a request from the client,
- Immediately before the database was queried for the XML transformation,
- After the transformation was complete,
- After the XQEngine had executed its query, and
- Just before the front servlet displayed the output on the screen.

In the following tables the 'Database to XML' column records the time taken for the transformation of the relational data into XML. The 'XQuery Engine' column records the time spent querying the generated XML document. Finally, the 'Data Transfer' column records the time interval from when the front servlet is activated to when the database transformation is started. This is added to the time taken for the results of the query to be transmitted back to the front servlet.

There were four queries used in the tests. For more information about creating XQueries see appendix D.

The first query:

```
//MoveRequestList
```

returns every movement request in the database. This produces the same result to the current CTL web services. The second query:

```
for $i in //RequestToMove where $i/@requestID [. &= 'test03' ] return $i
```

returns only the movement request that has a requestID of 'test03'. The third query:

```
for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i
```

returns the movement request where the first name of one of the people in the organization involved is Ron. Finally, the last query:

```
for $i in //Person where $i/First [. &= 'Ron'] return $i
```

only returns Ron's details, not a whole movement request.

'First time queried' indicates that this was the first time that a query was executed after the server (Tomcat) and browser (Internet Explorer) had been started. 'Subsequent queries' were those that were executed after an identical query had already been performed. The performance difference between the first and subsequent requests is because the application server that runs the servlets initialises various objects in the system to process the request. These objects stay initialised after execution and therefore do not need re-initialisation for the second and subsequent requests shortening the execution time.

This section is split into three parts, SQL Server version, Oracle Version and a summary of results. The SQL Server and Oracle sections detail the performance metrics collected on the two different databases. The performance tests manipulated two variables, the network bandwidth and the number of records contained in the database. The network bandwidth was altered to indicate the differences in time it would take to send and receive responses if the system was used over either intermittent or poor network bearers. The two networks it was tested on was either an 100Mbps local area network or over a 56Kbps dial-up modem. The summary section at the end of this chapter summarises the performance results.

3.1 SQL Server Version

Test performed on an instance of Microsoft SQL Server 2000.

3.1.1 Local Network (100Mbps)

Tests were run over the local DSTO network, which is a 100Mbps local area network.

3.1.1.1 Small Data (10 records in database)

Table 3-1: SQL Server Database, Local Network, Small Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	2534 ms	631 ms	1882 ms	5047 ms
Subsequent Queries	//MoveRequestList	991 ms	451 ms	89 ms	1531 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	741 ms	120 ms	45 ms	906 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	711 ms	70 ms	31 ms	812 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	941 ms	70 ms	520 ms	1531 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	531 ms	70 ms	56 ms	657 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	611 ms	60 ms	501 ms	1172 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	611 ms	60 ms	17 ms	688 ms

As shown in Table 3-1, when only 10 records are in the database, most queries executed in around one second. The first query, when executed for the first time, took 5 seconds because of the large result size. In all queries, more than half the total time was consumed by the database to XML conversion. In each query the XQuery Engine took only about one tenth of the total time.

3.1.1.2 Large Data (100 records in database)

Table 3-2: SQL Server Database, Local Network, Large Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	4607 ms	751 ms	9845 ms	15203 ms
Subsequent Queries	//MoveRequestList	4997 ms	751 ms	6690 ms	12438 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	4086 ms	941 ms	67 ms	5094 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	4276 ms	500 ms	37 ms	4813 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	4196 ms	511 ms	496 ms	5203 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	4086 ms	521 ms	34 ms	4641 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	4076 ms	511 ms	491 ms	5078 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	4256 ms	1032 ms	9 ms	5297 ms

As shown in Table 3-2, although there was ten times as much data in the database as the previous test, the total time for the query was only increased by a factor of five, with the last three queries only taking approximately five seconds each to complete. Once again, the first query took considerably longer to execute, at 15 seconds for the first time queried.

3.1.2 Dial Up Network (56Kbps)

Lists test that were conducted over a dial-up connection using a 57Kbps modem.

3.1.2.1 *Small Data (10 records in database)*

Table 3-3: SQL Server, Dial up connection, Small Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	590 ms	91 ms	94426 ms	95107 ms
Subsequent Queries	//MoveRequestList	601 ms	541 ms	90249 ms	91391 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	441 ms	70 ms	7951 ms	8462 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	601 ms	60 ms	6858 ms	7519 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	501 ms	70 ms	6540 ms	7111 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	601 ms	50 ms	6218 ms	6869 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	581 ms	50 ms	2254 ms	2885 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	521 ms	70 ms	1205 ms	1796 ms

Predictably, the majority of the total time in this test was taken up by transferring the data over the modem, as shown in Table 3-3. The smaller the data was, the faster the results were returned because the transfer did not take as long.

3.1.2.2 Large Data (100 records in database)

Table 3-4:SQL Server, Dial up connection, Small Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	8753 ms	1712 ms	436207 ms	446672 ms
Subsequent Queries	//MoveRequestList	4997 ms	741 ms	432412 ms	438150 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	4166 ms	511 ms	4987 ms	9664 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	4877 ms	500 ms	4978 ms	10355 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	4957 ms	501 ms	4877 ms	10335 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	4406 ms	561 ms	3546 ms	8513 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	4296 ms	481 ms	2093 ms	6870 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	5157 ms	491 ms	982 ms	6630 ms

Once again, a large amount of time was taken up in the transfer of data (see Table 3-4). Along with the data transfer, the database to XML conversion also took a substantial amount of time because of the large number of records in the database.

3.2 Oracle Version

3.2.1 Local Network (100Mbps)

Just as with the SQL Server version, tests were run over a 100Mbps local area network with the same machine as in the SQL Server tests.

3.2.1.1 *Small Data (10 records in database)*

Table 3-5: Oracle Database, Local Network, Small Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	3936 ms	450 ms	1317 ms	5703 ms
Subsequent Queries	//MoveRequestList	1582 ms	381 ms	287 ms	2250 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	3615 ms	451 ms	1075 ms	5141 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	2053 ms	60 ms	43 ms	2156 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	3575 ms	461 ms	1057 ms	5093 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	1943 ms	350 ms	51 ms	2344 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	3495 ms	441 ms	1001 ms	4937 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	2053 ms	60 ms	28 ms	2141 ms

As shown in Table 3-5, the first thing that becomes apparent about the Oracle version is that the Database to XML transformation is much slower in comparison to the SQL Server version. The transformation takes two to three times as long, stretching the total time to around five seconds, which seems to be a surprisingly long time for only ten records.

3.2.1.2 *Large Data (100 records in Database)*

Table 3-6: Oracle Database, Local Network, Large Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	27239 ms	1322 ms	9282 ms	37843 ms
Subsequent Queries	//MoveRequestList	21001 ms	721 ms	8855 ms	30577 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	20530 ms	470 ms	499 ms	21499 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	19979 ms	931 ms	43 ms	20953 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	19508 ms	481 ms	526 ms	20515 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	20018 ms	531 ms	60 ms	20609 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	19888 ms	461 ms	510 ms	20859 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	20499 ms	461 ms	40 ms	21000 ms

With 100 records in the database, each query is taking more than 20 seconds to process, as shown in Table 3-6. This is almost entirely due to the database to XML transformation, which is more than four times slower than the SQL server version.

3.2.2 Dial Up Network (56Kbps)

As with the SQL Server version, a 56k modem connection dialling in from a remote location was tested.

3.2.2.1 *Small Data (10 records in Database)*

Table 3-7: Oracle Database, Dial up connection, Small Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	1402 ms	60 ms	36392 ms	37854 ms
Subsequent Queries	//MoveRequestList	1192 ms	50 ms	34690 ms	35932 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	1132 ms	40 ms	6629 ms	7801 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	1232 ms	50 ms	5928 ms	7210 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	1252 ms	50 ms	4867 ms	6169 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	1172 ms	40 ms	3775 ms	4987 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	1252 ms	30 ms	2324 ms	3606 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	1462 ms	30 ms	1561 ms	2553 ms

As with the SQL server version, the data transfer was the part of the process that took the most time over a 56K modem (see Table 3-7). The database to XML transformation was once again slower than it was using SQL server.

3.2.2.2 *Large Data (100 records in Database)*

Table 3-8: Oracle Database, Dial up connection, Large Data

	Query	Database to XML	XQuery Engine	Data Transfer	Total Time
First Time Queried	//MoveRequestList	19528 ms	1332 ms	428586 ms	449446 ms
Subsequent Queries	//MoveRequestList	15342 ms	541 ms	426223 ms	442116 ms
First Time Queried	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	14441 ms	390 ms	7461 ms	22292 ms
Subsequent Queries	for $i in //RequestToMove where $i/@requestID [. &= 'test03'] return $i	14781 ms	351 ms	5377 ms	20509 ms
First Time Queried	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	14962 ms	701 ms	6038 ms	21701 ms
Subsequent Queries	for $i in //RequestToMove where $i/Organization/Person/First [. &= 'Ron'] return $i	14240 ms	681 ms	3776 ms	18697 ms
First Time Queried	for $i in //Person where $i/First [. &= 'Ron'] return $i	14440 ms	671 ms	3055 ms	18166 ms
Subsequent Queries	for $i in //Person where $i/First [. &= 'Ron'] return $i	14691 ms	330 ms	1093 ms	16114 ms

As Table 3-8 shows, these results are similar to the SQL Server version, with the data transfer taking the most time, although once again, the transformation to XML was slower using Oracle.

3.3 Summary of results

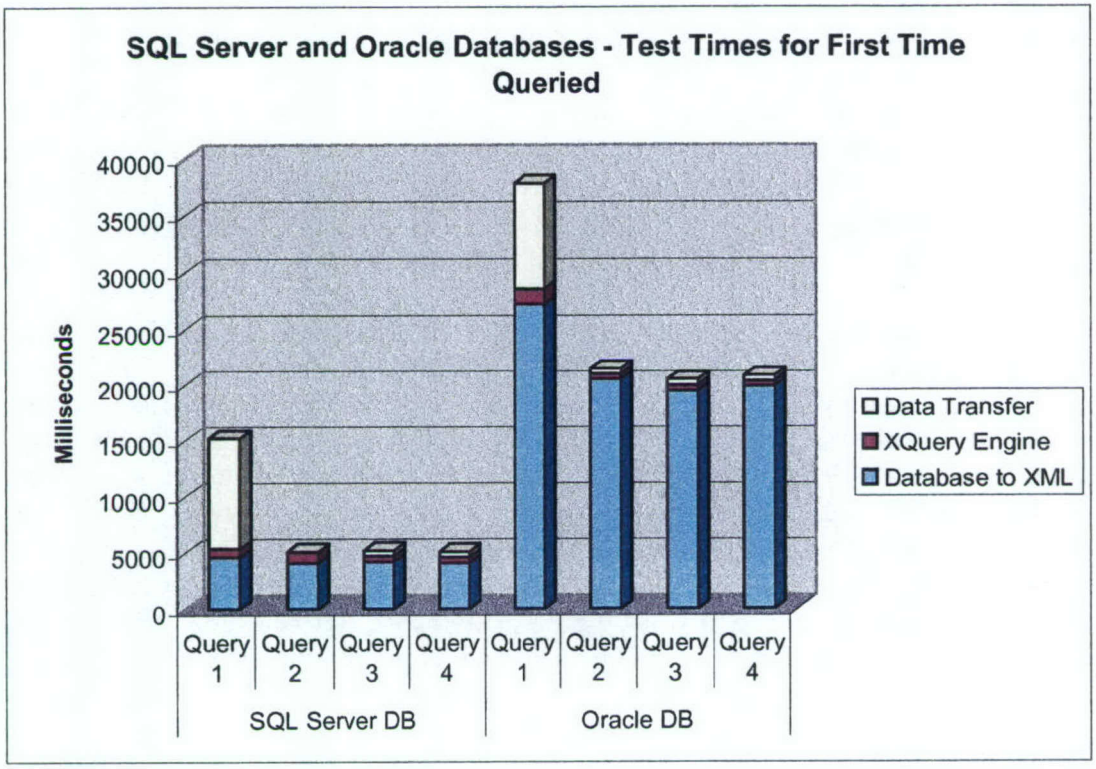


Figure 4: Graph of test times comparing SQL Server and Oracle Databases over 100Mbs network with 100 records in the Databases

A number of trends in the test results are apparent. Firstly, over a reasonably fast connection, the bottleneck of the process can be attributed to the transformation from relational data to XML. This is especially true when querying an Oracle database, which took up to four times longer to complete the transformation when compared to the SQL Server database (see Figure 4). By comparison the XQuery Engine takes very little time to get the query results from the XML data once it has been transformed. Over a slower connection the size of the results begins to have a more significant effect on the overall performance, and returning all the records over a modem can take several minutes.

4. Conclusion

The XQuery prototype has achieved what it set out to demonstrate, namely that XQuery is an effective and useful tool for returning selected parts of an XML document. Although there are still some drawbacks associated with the implementation of the prototype, overall the project has been a success.

By far the largest issue yet to be resolved is how to transform the relational data residing in the database into XML. Currently the transformation is 'hard-coded', so that if the structure of the database or XML changes, the code will need to be updated. A way of mapping the transformation between the database and XML structures must be found to improve the maintainability of the system.

Aside from considerations of flexibility, the current transformation process is simply too slow, especially with large amounts of data and using an Oracle database. Over a network connection, querying an Oracle database, the transformation process took about 90% of the total time it to return the results of a query. While there may be a number of ways to cut down this time, the main reason for the delay is that the prototype extracts *all* the information from the database, transforming it into XML and *then* queries it. This process is an inherently inefficient way of querying the data especially when databases provide querying capabilities. A better method would be to allow the database to apply the XQuery statements on the data contained within although no current database allows this on relational data. It is recommended that future research be conducted into querying databases and providing relational to XML translations that are quicker and more efficient.

Apart from this issue, a number of positives have come out of the development of the prototype. The XQuery implementation, XQEngine, has met all requirements and has proven to be fast and powerful. The use of the XQuery language enables flexible and precise searches, so that the data returned is exactly what the user requires.

The XQuery Prototype has demonstrated that it is possible to utilise XQuery to search through transformed relational data, and that there are significant advantages to be gained by using this method. There remain some faults and omissions in the prototype that would need to be rectified before a true web service for CTL could be established, but the work done so far will provide a solid basis for future implementation and exploration of this new technology. It is therefore recommended that CTL integrate the XQuery capability to provide querying functionality on all web services provided.

References

1. G. Clare, M. Schenk (April 2001), *Coalition Theatre Logistics Requirement Study*, Produced by SMS Consulting for the Defence Knowledge Improvement Team

2. Egon Kuster (2002), *CTL Architecture Overview (Draft)*, DSTO, Command and Control Division

3. Egon Kuster (December 2002), *DSTO CTL Web Site*, http://ctl-web.dsto.defence.gov.au/CTL/

4. W3C (May 2000), *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3.org/TR/SOAP/

5. W3C (March 2001), *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/wsdl

6. Jason Hunter and Brett McLaughlin (February 2003), *JDOM Homepage*, http://www.jdom.org/

7. W3C (November 1999), *XML Path Language (XPath)*, http://www.w3.org/TR/xpath

8. W3C (November 2002), *XQuery 1.0: An XML Query Language*, http://www.w3.org/TR/xquery/

9. Howard Katz (September 2002), *DeveloperWorks: XML Zone: An Introduction to XQuery*, http://www-106.ibm.com/developerworks/library/x-xquery.html

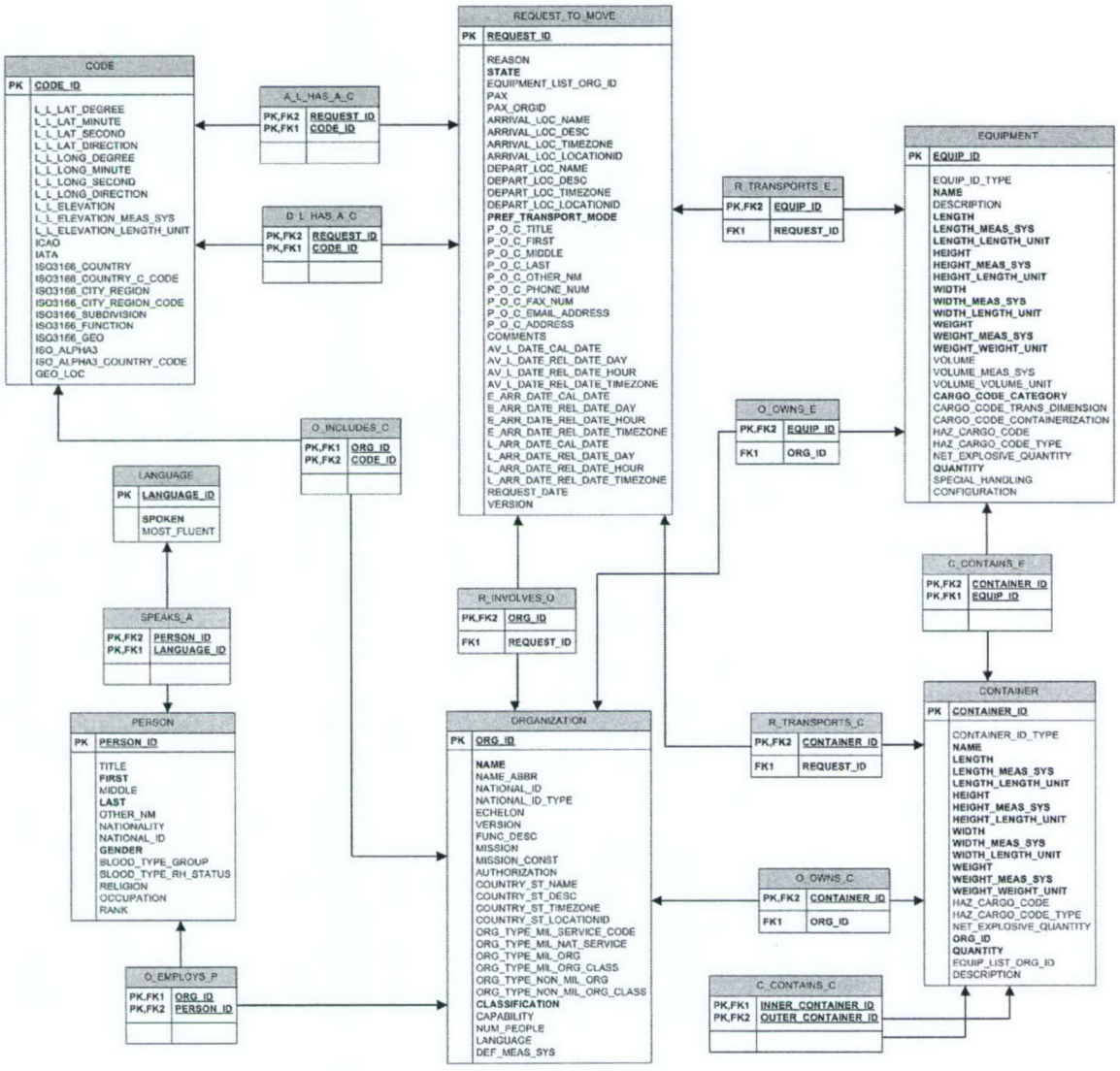
10. Howard Katz (December 2002), *XQEngine*, http://www.fatdog.com/

11. Sun Microsystems (December 2002), *JDBC Technology*, http://java.sun.com/products/jdbc/

12. Directorate of Military Strategy and Strategic Doctrine (October 2001), *ADP 00.03 Coalition Operations – Draft 04*, Australian Defence Doctrine Publication

Appendix A: Database Schema

This is the schema for the database within which the relational data for the movement requests was stored.



Appendix B: Web Service WSDL

A Web Service Description Language (WSDL) file describes a web service, providing information on the uses of the service and how it is called. This is the WSDL file for the XQuery Prototype.

```
<?xml version="1.0"?>
<wsdl:definitions name="xQueryPrototype"
    targetNamespace="http://shapecharge.dsto.defence.gov.au/schemas/XQueryPrototype.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <wsdl:message name="getOracleXQResultsRequest">
        <wsdl:part name="query" type="xs:string"/>
    </wsdl:message>

    <wsdl:message name="getOracleXQResultsResponse">
        <wsdl:part name="results" type="xs:string"/>
    </wsdl:message>

    <wsdl:message name="getMSSQLXQResultsRequest">
        <wsdl:part name="query" type="xs:string"/>
    </wsdl:message>

    <wsdl:message name="getMSSQLXQResultsResponse">
        <wsdl:part name="results" type="xs:string"/>
    </wsdl:message>

    <wsdl:portType name="listResults">
        <wsdl:operation name="getOracleXQResults">
            <wsdl:input message="getOracleXQResultsRequest"/>
            <wsdl:output message="getOracleXQResultsResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getMSSQLXQResults">
            <wsdl:input message="getMSSQLXQResultsRequest"/>
            <wsdl:output message="getMSSQLXQResultsResponse"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding type="listResults" name="xQueryPrototypeSoap">
        <soap:binding style="document"  transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="getOracleXQResults">
            <soap:operation style="document"
                soapAction="http://shapecharge:80/query_test/servlet/OracleBackServlet"/>
            <wsdl:input>
                <soap:body parts="body" use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body parts="body" use="literal" encodingStyle="http//schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getMSSQLXQResults">
            <soap:operation style="document"
                soapAction="http://shapecharge2:8080/query_test/servlet/SQLServerBackServlet"/>
            <wsdl:input>
                <soap:body parts="body" use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body parts="body" use="literal" encodingStyle="http//schemas.xmlsoap.org/soap/encoding/"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Appendix C: Sample XML Data

The following example of a movement request list document contains three movement requests.

```
<MoveRequestList>
   <RequestToMove requestID="movemf18326a9a50000">
      <Reason>To move the Med Coy 5 CSSB into the coalition</Reason>
      <State>PENDING</State>
      <Organization orgID="organf18326a9b90007">
         <Name>Med Coy 5 CSSB</Name>
         <Echelon>COMPANY</Echelon>
         <Version>1.5</Version>
         <FuncDesc>Medical Company</FuncDesc>
         <CountrySt>
            <Name>Queenbeyan</Name>
            <Desc>Queenbeyan, NSW, Australia</Desc>
            <TimeZone>+10</TimeZone>
         </CountrySt>
         <OrgType>
            <Military milOrgClass="COMBAT SUPPORT">
               <ServiceCode>LAND</ServiceCode>
               <NationalService>ARMY</NationalService>
               <MilitaryOrg>MEDICAL</MilitaryOrg>
            </Military>
         </OrgType>
         <Classification>COALITION_PARTNER</Classification>
         <NumPeople>55</NumPeople>
         <DefMeasSys>Metric</DefMeasSys>
         <Equipment equipID="equipf18326a9cd0013">
            <Name>LR 110</Name>
            <Length measSys="METRIC" lengthUnit="METER">4.83</Length>
            <Height measSys="METRIC" lengthUnit="METER">2.04</Height>
            <Width measSys="METRIC" lengthUnit="METER">2.06</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">2250.0</Weight>
            <CargoCode>
               <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <Quantity>3</Quantity>
         </Equipment>
         <Equipment equipID="equipf18326a9cd0014">
            <Name>Ambulance</Name>
            <Length measSys="METRIC" lengthUnit="METER">6.0</Length>
            <Height measSys="METRIC" lengthUnit="METER">2.59</Height>
            <Width measSys="METRIC" lengthUnit="METER">2.16</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">5800.0</Weight>
            <CargoCode>
               <Category>VEHICLES NON-SELF DEPLOYING</Category>
            </CargoCode>
            <Quantity>4</Quantity>
         </Equipment>
         <Equipment equipID="equipf18326a9cd0015">
            <Name>Unimog</Name>
            <Length measSys="METRIC" lengthUnit="METER">6.75</Length>
            <Height measSys="METRIC" lengthUnit="METER">3.13</Height>
            <Width measSys="METRIC" lengthUnit="METER">2.47</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">6800.0</Weight>
            <CargoCode>
               <Category>VEHICLES NON-SELF DEPLOYING</Category>
            </CargoCode>
            <Quantity>4</Quantity>
         </Equipment>
         <Equipment equipID="equipf18326a9cd0016">
            <Name>Bx Lithium Battery</Name>
            <Length measSys="METRIC" lengthUnit="METER">0.42</Length>
```

```
            <Height measSys="METRIC" lengthUnit="METER">0.25</Height>
            <Width measSys="METRIC" lengthUnit="METER">0.15</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">13.5</Weight>
            <CargoCode>
                <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <HazCargo codeType="UN">9</HazCargo>
            <NetExplosiveQuantity>0.0</NetExplosiveQuantity>
            <Quantity>5</Quantity>
        </Equipment>
        <Equipment equipID="equipf18326a9cd0017">
            <Name>30KVA</Name>
            <Length measSys="METRIC" lengthUnit="METER">0.0</Length>
            <Height measSys="METRIC" lengthUnit="METER">0.0</Height>
            <Width measSys="METRIC" lengthUnit="METER">0.0</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">0.0</Weight>
            <CargoCode>
                <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <HazCargo codeType="UN">9</HazCargo>
            <NetExplosiveQuantity>0.0</NetExplosiveQuantity>
            <Quantity>2</Quantity>
        </Equipment>
        <Equipment equipID="equipf18326a9cd0018">
            <Name>16KVA</Name>
            <Length measSys="METRIC" lengthUnit="METER">1.57</Length>
            <Height measSys="METRIC" lengthUnit="METER">1.17</Height>
            <Width measSys="METRIC" lengthUnit="METER">0.88</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">450.0</Weight>
            <CargoCode>
                <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <HazCargo codeType="UN">9</HazCargo>
            <NetExplosiveQuantity>0</NetExplosiveQuantity>
            <Quantity>5</Quantity>
        </Equipment>
        <Equipment equipID="equipf18326a9cd0019">
            <Name>Gas Cylinder</Name>
            <Length measSys="METRIC" lengthUnit="METER">0.5</Length>
            <Height measSys="METRIC" lengthUnit="METER">1.2</Height>
            <Width measSys="METRIC" lengthUnit="METER">0.5</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">35.0</Weight>
            <CargoCode>
                <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <HazCargo codeType="UN">9</HazCargo>
            <NetExplosiveQuantity>4</NetExplosiveQuantity>
            <Quantity>6</Quantity>
        </Equipment>
        <Container containerID="contaf18860840d0000" containerType="ISO Pallet">
            <Length measSys="METRIC" lengthUnit="METER">1.1</Length>
            <Height measSys="METRIC" lengthUnit="METER">0.0</Height>
            <Width measSys="METRIC" lengthUnit="METER">1.1</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">1000.0</Weight>
            <OrgID>organf18326a9b90007</OrgID>
            <Quantity>8</Quantity>
        </Container>
    </Organization>
    <ArrivalLoc>
        <Name>Tindoro</Name>
        <Timezone>+10</Timezone>
    </ArrivalLoc>
    <DepartLoc>
        <Name>Sydney</Name>
        <Timezone>+10</Timezone>
    </DepartLoc>
    <PrefTransportMode>ANY</PrefTransportMode>
    <PointOfContact>
        <Title>Captain</Title>
        <FirstNm>E.</FirstNm>
```

```
            <LastNm>Collins</LastNm>
            <PhoneNum>(02) 6265 1003</PhoneNum>
            <Address>HMAS Harman, Queenbeyan, NSW, 2456, Australia</Address>
        </PointOfContact>
        <Comments>Exercise Only</Comments>
        <AvailLoadDate>
            <CalDate>2002-12-03</CalDate>
        </AvailLoadDate>
        <LatestArrivalDate>
            <CalDate>2002-12-06</CalDate>
        </LatestArrivalDate>
        <RequestDate>2002-08-13</RequestDate>
        <Version>1.5</Version>
    </RequestToMove>
    <RequestToMove requestID="movemf18326a9a50001">
        <Reason>I am not sure what the reason is :)</Reason>
        <State>PENDING</State>
        <Organization orgID="organf18326a9b90008">
            <Name>A Coy 2/17 RNSWR</Name>
            <Echelon>COMPANY</Echelon>
            <Version>1.5</Version>
            <FuncDesc>Not sure what this does talk to the Point of Contact</FuncDesc>
            <CountrySt>
                <Name>Canberra</Name>
                <Desc>Canberra City, Canberra, ACT, Australia</Desc>
                <TimeZone>+10</TimeZone>
            </CountrySt>
            <OrgType>
                <Military milOrgClass="COMBAT">
                    <ServiceCode>LAND</ServiceCode>
                    <NationalService>ARMY</NationalService>
                    <MilitaryOrg>MISC, CMBT, CMBT SPT, CMBT SERV SPT, UNIT HQ</MilitaryOrg>
                </Military>
            </OrgType>
            <Classification>COALITION_PARTNER</Classification>
            <NumPeople>117</NumPeople>
            <DefMeasSys>Metric</DefMeasSys>
            <Equipment equipID="equipf18326a9cd0020">
                <Name>LR 110</Name>
                <Length measSys="METRIC" lengthUnit="METER">4.83</Length>
                <Height measSys="METRIC" lengthUnit="METER">2.04</Height>
                <Width measSys="METRIC" lengthUnit="METER">2.06</Width>
                <Weight measSys="METRIC" weightUnit="KILOGRAM">2250.0</Weight>
                <CargoCode>
                    <Category>OTHER NON-VEHICLE</Category>
                </CargoCode>
                <Quantity>5</Quantity>
            </Equipment>
            <Container containerID="contaf18860840d0001" containerType="ISO Pallet">
                <Length measSys="METRIC" lengthUnit="METER">1.1</Length>
                <Height measSys="METRIC" lengthUnit="METER">0.0</Height>
                <Width measSys="METRIC" lengthUnit="METER">1.1</Width>
                <Weight measSys="METRIC" weightUnit="KILOGRAM">1000.0</Weight>
                <OrgID>organf18326a9b90008</OrgID>
                <Quantity>10</Quantity>
                <Desc>This is a description about the pallet and its contents</Desc>
            </Container>
        </Organization>
        <ArrivalLoc>
            <Name>Sydney Pymble Bks</Name>
            <Timezone>+10</Timezone>
        </ArrivalLoc>
        <DepartLoc>
            <Name>Arden Street, Canberra</Name>
            <Timezone>+10</Timezone>
        </DepartLoc>
        <PrefTransportMode>ORGANIC_LAND</PrefTransportMode>
        <PointOfContact>
            <Title>Capt</Title>
            <FirstNm>E.</FirstNm>
```

```
            <LastNm>Collins</LastNm>
            <PhoneNum>(02) 6265 1003</PhoneNum>
            <Address>1 Arden St, Canberra City, Canberra, ACT, 2600</Address>
        </PointOfContact>
        <Comments>This is Exercise Only. Bn to provide 3 x buses and 3 x macks for stores and pers
            from unit 1st line. Bn concentration in Sydney Bks</Comments>
        <AvailLoadDate>
            <CalDate>2002-12-01</CalDate>
        </AvailLoadDate>
        <LatestArrivalDate>
            <CalDate>2002-12-01</CalDate>
        </LatestArrivalDate>
        <RequestDate>2002-08-13</RequestDate>
        <Version>1.5</Version>
    </RequestToMove>
    <RequestToMove requestID="movemf18326a9a50002">
        <Reason>I am not sure what the reason is :)</Reason>
        <State>PENDING</State>
        <Organization orgID="organf18326a9b90009">
            <Name>A Coy 2/17 RNSWR</Name>
            <Echelon>COMPANY</Echelon>
            <Version>1.5</Version>
            <FuncDesc>Not sure what this does talk to the Point of Contact</FuncDesc>
            <CountrySt>
                <Name>Canberra</Name>
                <Desc>Canberra City, Canberra, ACT, Australia</Desc>
                <TimeZone>+10</TimeZone>
            </CountrySt>
            <OrgType>
                <Military milOrgClass="COMBAT">
                    <ServiceCode>LAND</ServiceCode>
                    <NationalService>ARMY</NationalService>
                    <MilitaryOrg>MISC, CMBT, CMBT SPT, CMBT SERV SPT, UNIT HQ</MilitaryOrg>
                </Military>
            </OrgType>
            <Classification>COALITION_PARTNER</Classification>
            <NumPeople>117</NumPeople>
            <DefMeasSys>Metric</DefMeasSys>
            <Equipment equipID="equipf18326a9cd0021">
                <Name>LR 110</Name>
                <Length measSys="METRIC" lengthUnit="METER">4.83</Length>
                <Height measSys="METRIC" lengthUnit="METER">2.04</Height>
                <Width measSys="METRIC" lengthUnit="METER">2.06</Width>
                <Weight measSys="METRIC" weightUnit="KILOGRAM">2250.0</Weight>
                <CargoCode>
                    <Category>OTHER NON-VEHICLE</Category>
                </CargoCode>
                <Quantity>5</Quantity>
            </Equipment>
            <Equipment equipID="equipf18326a9cd0022">
                <Name>0.5/1t</Name>
                <Length measSys="METRIC" lengthUnit="METER">3.15</Length>
                <Height measSys="METRIC" lengthUnit="METER">1.37</Height>
                <Width measSys="METRIC" lengthUnit="METER">1.57</Width>
                <Weight measSys="METRIC" weightUnit="KILOGRAM">255.0</Weight>
                <CargoCode>
                    <Category>VEHICLES NON-SELF DEPLOYING</Category>
                </CargoCode>
                <Quantity>5</Quantity>
            </Equipment>
            <Equipment equipID="equipf18326a9cd0023">
                <Name>Bx Lithium Battery</Name>
                <Length measSys="METRIC" lengthUnit="METER">0.42</Length>
                <Height measSys="METRIC" lengthUnit="METER">0.25</Height>
                <Width measSys="METRIC" lengthUnit="METER">0.15</Width>
                <Weight measSys="METRIC" weightUnit="KILOGRAM">13.5</Weight>
                <CargoCode>
                    <Category>OTHER NON-VEHICLE</Category>
                </CargoCode>
                <HazCargo codeType="UN">9</HazCargo>
```

```
            <Quantity>10</Quantity>
        </Equipment>
        <Equipment equipID="equipf18326a9cd0024">
            <Name>Bx Hexamine</Name>
            <Length measSys="METRIC" lengthUnit="METER">0.45</Length>
            <Height measSys="METRIC" lengthUnit="METER">0.28</Height>
            <Width measSys="METRIC" lengthUnit="METER">0.35</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">75.0</Weight>
            <CargoCode>
                <Category>OTHER NON-VEHICLE</Category>
            </CargoCode>
            <HazCargo codeType="UN">4</HazCargo>
            <NetExplosiveQuantity>0.0</NetExplosiveQuantity>
            <Quantity>50</Quantity>
        </Equipment>
        <Container containerID="contaf18860840d0002" containerType="ISO Pallet">
            <Length measSys="METRIC" lengthUnit="METER">1.1</Length>
            <Height measSys="METRIC" lengthUnit="METER">0.0</Height>
            <Width measSys="METRIC" lengthUnit="METER">1.1</Width>
            <Weight measSys="METRIC" weightUnit="KILOGRAM">1000.0</Weight>
            <OrgID>organf18326a9b90009</OrgID>
            <Quantity>10</Quantity>
            <Desc>This is a description about the pallet and its contents</Desc>
        </Container>
    </Organization>
    <ArrivalLoc>
        <Name>Tindoro</Name>
        <Timezone>+10</Timezone>
    </ArrivalLoc>
    <DepartLoc>
        <Name>Sydney</Name>
        <Timezone>+10</Timezone>
    </DepartLoc>
    <PrefTransportMode>AIR</PrefTransportMode>
    <PointOfContact>
        <Title>Capt</Title>
        <FirstNm>E.</FirstNm>
        <LastNm>Collins</LastNm>
        <PhoneNum>(02) 6265 1003</PhoneNum>
        <Address>1 Arden St, Canberra City, Canberra, ACT, 2600</Address>
    </PointOfContact>
    <Comments>This is Exercise Only.</Comments>
    <AvailLoadDate>
        <CalDate>2002-12-05</CalDate>
    </AvailLoadDate>
    <LatestArrivalDate>
        <CalDate>2002-12-07</CalDate>
    </LatestArrivalDate>
    <RequestDate>2002-08-13</RequestDate>
    <Version>1.5</Version>
</RequestToMove>
</MoveRequestList>
```

Appendix D: Sample XQuery Statements

The simplest XQuery statement one can enter is to return all of the data in the database, that is, all of the movement requests in the database. To do this we select the root element from the generated XML document:

//MoveRequestList

This statement simply tells the XQEngine to return all <MoveRequestList> elements and their contents. The two forward slashes indicate that this is the topmost element, or starting element, for the search, something that will become more meaningful in more advanced searches.

The criterion for selecting a movement request out of the list can be based on the content of any element or attribute within the request. The following query selects a movement request based on the number of passengers to be transported:

for $i in //RequestToMove where $i/Pax [. &= '55'] return $i

The first part of the query declares a variable, $i, and gives it the value //RequestToMove. The second part of the query asks the XQEngine to return $i when the Pax element in $i has a value of '55'. The '[. &= 'value']' syntax is XQuery's way of specifying the value (or values) to search for. Thus the search

for $i in //RequestToMove where $i/Reason [. &= 'conflict medicine'] return $i

Would return all movement requests where the Reason element contained the words 'conflict' and 'medicine'. Note that the XQuery language, like XML is case sensitive, so care must be taken when describing elements and attributes to search, as well as the values to be searched for.

One advantage of using XQuery is that you do not necessarily need to know the exact location of an element in the structure of the XML document to search it. For example, the Person element resides in the Organization element, which in turn is a child to the RequestToMove element. The following search will return data about a person based on their last name.

for $i in //RequestToMove/Organization/Person where $i/Last [. &= 'Smith'] return $i

The following search will return exactly the same results:

for $i in //Person where $i/Last [. &= 'Smith'] return $i

However, neither of these queries would yield the same results as the following search:

for $i in //RequestToMove where $i/Organization/Person/Last [. &= 'Smith'] return $i

This query returns all RequestToMove elements and everything inside them where the last name of a person is 'Smith', compared to only the Person element as in the previous two queries. This demonstrates that it is the value of the variable that determines how much of the document is returned. Apart from a general knowledge of the structure of the XML document, this is all the information required to perform effective queries of a movement request list. For more information on XQuery, and a quick tutorial, visit the IBM DeveloperWorks tutorial on XQuery at http://www-106.ibm.com/developerworks/library/x-xquery.html .

DISTRIBUTION LIST

XQuery Engine Prototype
Coalition Theatre Logistics (CTL)
Advanced Concept Technology Demonstration (ACTD)

Egon Kuster and Andrew Roff

AUSTRALIA

DEFENCE ORGANISATION

	No. of copies
Task Sponsor	
DGICD	1
S&T Program	
Chief Defence Scientist	shared copy
FAS Science Policy	
AS Science Corporate Management	
Director General Science Policy Development	
Counsellor Defence Science, London	Doc Data Sheet
Counsellor Defence Science, Washington	Doc Data Sheet
Scientific Adviser to MRDC, Thailand	Doc Data Sheet
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc Data Sheet & Distribution List
Scientific Adviser - Army	Doc Data Sheet & Distribution List
Air Force Scientific Adviser	Doc Data Sheet & Distribution List
Scientific Adviser to the DMO M&A	Doc Data Sheet & Distribution List
Scientific Adviser to the DMO ELL	Doc Data Sheet & Distribution List
Information Sciences Laboratory	
Chief Command & Control Division	Doc Data Sheet
Research Leader Command Decision Environments Branch	1
Research Leader Information Enterprises Branch	1
Research Leader Joint Command Analysis Branch	Doc Data Sheet
Research Leader Intelligence Information Branch	Doc Data Sheet
Head Human Systems Integration	Doc Data Sheet
Head Information Exploitation	Doc Data Sheet
Head Effects-Based Modelling and Analysis	Doc Data Sheet
Head Information Systems	1
Head Distributed Enterprises	Doc Data Sheet
Head Joint Operations Analysis and Support	Doc Data Sheet
Head Command Concepts and Architectures	Doc Data Sheet

Head Command Process Integration and Analysis	Doc Data Sheet
Head Intelligence Analysis	Doc Data Sheet
Dr Andrew Au, CCA Fernhill (Task Manager)	1
Egon Kuster (Author)	2
Publications and Publicity Officer, C2D/EOC2D	1 shared copy

DSTO Library and Archives

Library Edinburgh	1
Defence Archives	1
Library Canberra	Doc Data Sheet

Capability Development Group

Director General Maritime Development	Doc Data Sheet
Director General Land Development	1
Director General Capability and Plans	Doc Data Sheet
Assistant Secretary Investment Analysis	Doc Data Sheet
Director Capability Plans and Programming	Doc Data Sheet
Director Trials	Doc Data Sheet
Australian CTL Project Manager – Mr Selby Dyer	1

Chief Information Officer Group

Deputy CIO	Doc Data Sheet
Director General Information Policy and Plans	Doc Data Sheet
AS Information Strategy and Futures	Doc Data Sheet
AS Information Architecture and Management	Doc Data Sheet
Director General Australian Defence Simulation Office	Doc Data Sheet
Director General Information Services	Doc Data Sheet

Strategy Group

Director General Military Strategy	Doc Data Sheet
Director General Preparedness	Doc Data Sheet
Assistant Secretary Strategic Policy	Doc Data Sheet
Assistant Secretary Governance and Counter-Proliferation	Doc Data Sheet

Navy

Maritime Operational Analysis Centre, Building 89/90 NSW Garden Island Sydney	Doc Data Sht & Distribution List
Director General Navy Capability, Performance and Plans, Navy Headquarters	Doc Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters	Doc Data Sheet

Air Force

SO (Science) - Headquarters Air Combat Group, RAAF Base, Williamtown NSW 2314	Doc Data Sht & Exec Summary

Army

SO (Science) - Land Headquarters (LHQ), Victoria Barracks NSW — Doc Data & Exec Summ

SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Enoggera QLD — Doc Data Sheet

Joint Operations Command

Director General Joint Operations	Doc Data Sheet
Chief of Staff Headquarters Joint Operations Command	Doc Data Sheet
Commandant ADF Warfare Centre	Doc Data Sheet
Director General Strategic Logistics	Doc Data Sheet

Intelligence and Security Group

DGSTA Defence Intelligence Organisation	1
Manager, Information Centre, Defence Intelligence Organisation	1 (PDF)
Assistant Secretary Capability Provisioning	Doc Data Sheet
Assistant Secretary Capability and Systems	Doc Data Sheet

Assistant Secretary Corporate, Defence Imagery and Geospatial Organisation Doc Data Sheet

Defence Materiel Organisation

Deputy CEO	Doc Data Sheet
Head Aerospace Systems Division	Doc Data Sheet
Head Maritime Systems Division	Doc Data Sheet
Chief Joint Logistics Command	Doc Data Sheet
Head Materiel Finance	Doc Data Sheet

Defence Libraries

Library Manager, DLS-Canberra	1
Library Manager, DLS - Sydney West	Doc Data Sheet

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy Library	1
Head of Aerospace and Mechanical Engineering	1
Hargrave Library, Monash University	Doc Data Sheet
Librarian, Flinders University	1

OUTSIDE AUSTRALIA

INTERNATIONAL DEFENCE INFORMATION CENTRES

US Defense Technical Information Center	2
UK Dstl Knowledge Services	2
Canada Defence Research Directorate R&D Knowledge & Information Management (DRDKIM)	1
NZ Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

SPARES	5

Total number of copies: Printed 36 PDF 1 = 37

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

XQuery Engine Prototype
Coalition Theatre Logistics (CTL)
Advanced Concept Technology Demonstrator (ACTD)

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Egon Kuster and Andrew Roff

5. CORPORATE AUTHOR

Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-0577	AR-013-071	Technical Note	October 2004

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
9505-25-44	01/307	DGICD	25	9

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0577.pdf

14. RELEASE AUTHORITY

Chief, Command and Control Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DEFTEST DESCRIPTORS

Joint Military Activities, Joint Operations, Logistics Information Systems, Computer Network Architecture

19. ABSTRACT

During the architectural design of the Coalition Theatre Logistics (CTL) Advanced Concept Technology Demonstrator (ACTD) it was identified that a data query capability that could operate over XML-based web services was required. This document outlines how a new technology called XQuery could provide this XML-based query capability over web service oriented communication. Also outlined is a possible solution along with a discussion of its limitations and capabilities. A successful implementation of the XQuery engine was developed with performance metrics and architectural designs of the implemented system included within.

TECHNICAL NOTE DSTO-TN-0577 AR-013-071 OCTOBER 2004

Australian Government

Department of Defence

Defence Science and
Technology Organisation

INFORMATION SCIENCES LABORATORY
PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111
AUSTRALIA. TELEPHONE (08) 8259 5555